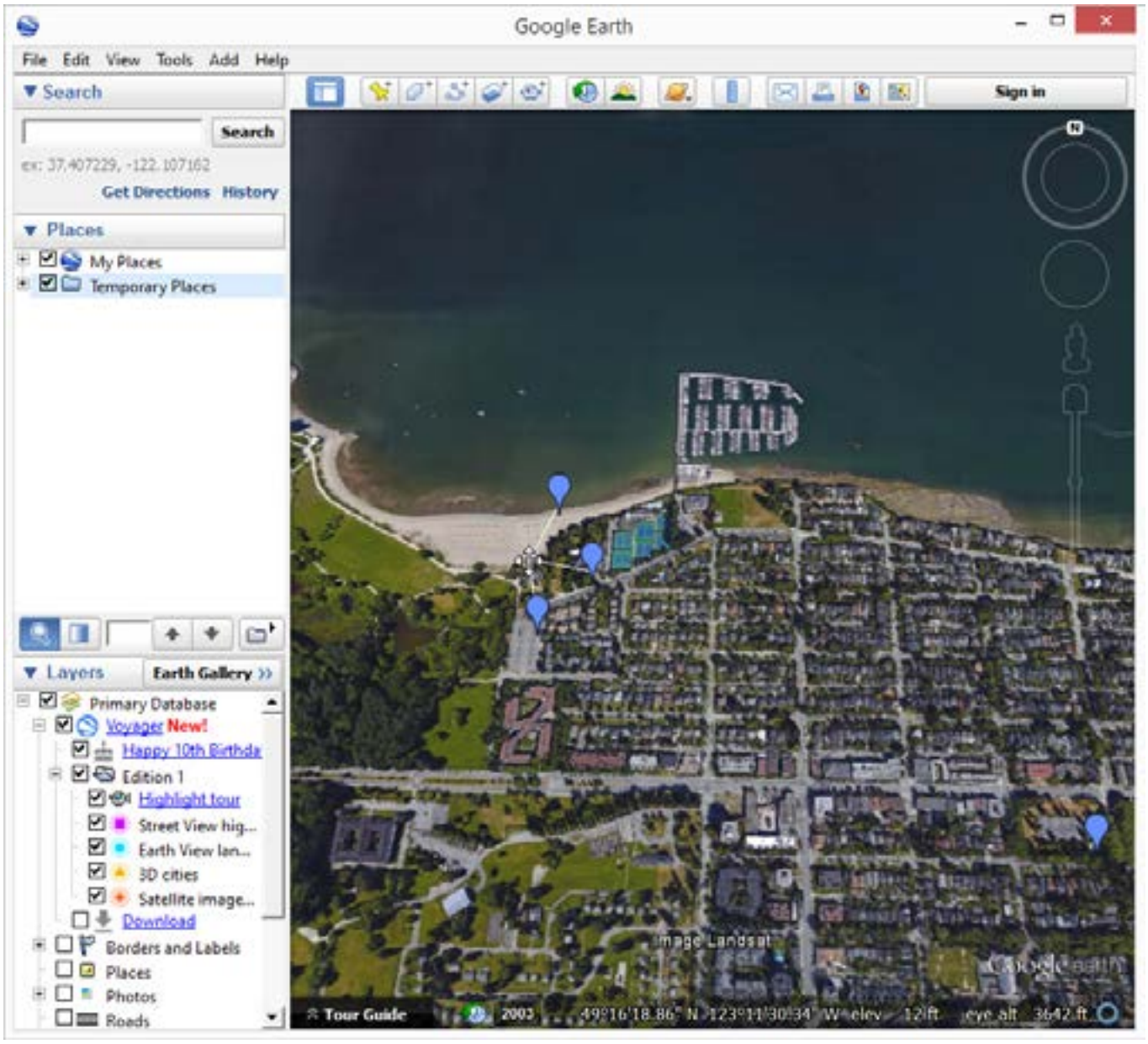
- Safe Software Presents -

Top 10 Geometry Struggles and How To Fix Them

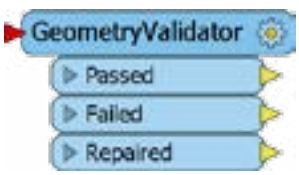
Part of FME's job description is to **identify and fix data problems**. This means we deal with a lot of bad data here at Safe. Compliancy failures, horrifying structural issues, values that are just plain wrong, null and missing pieces... and, of course, **messy geometry**. It's amazing how creative geometry can get, showing up in any way imaginable except the way you intended.

Here are the 10 most common struggles our users have when working with geometry.

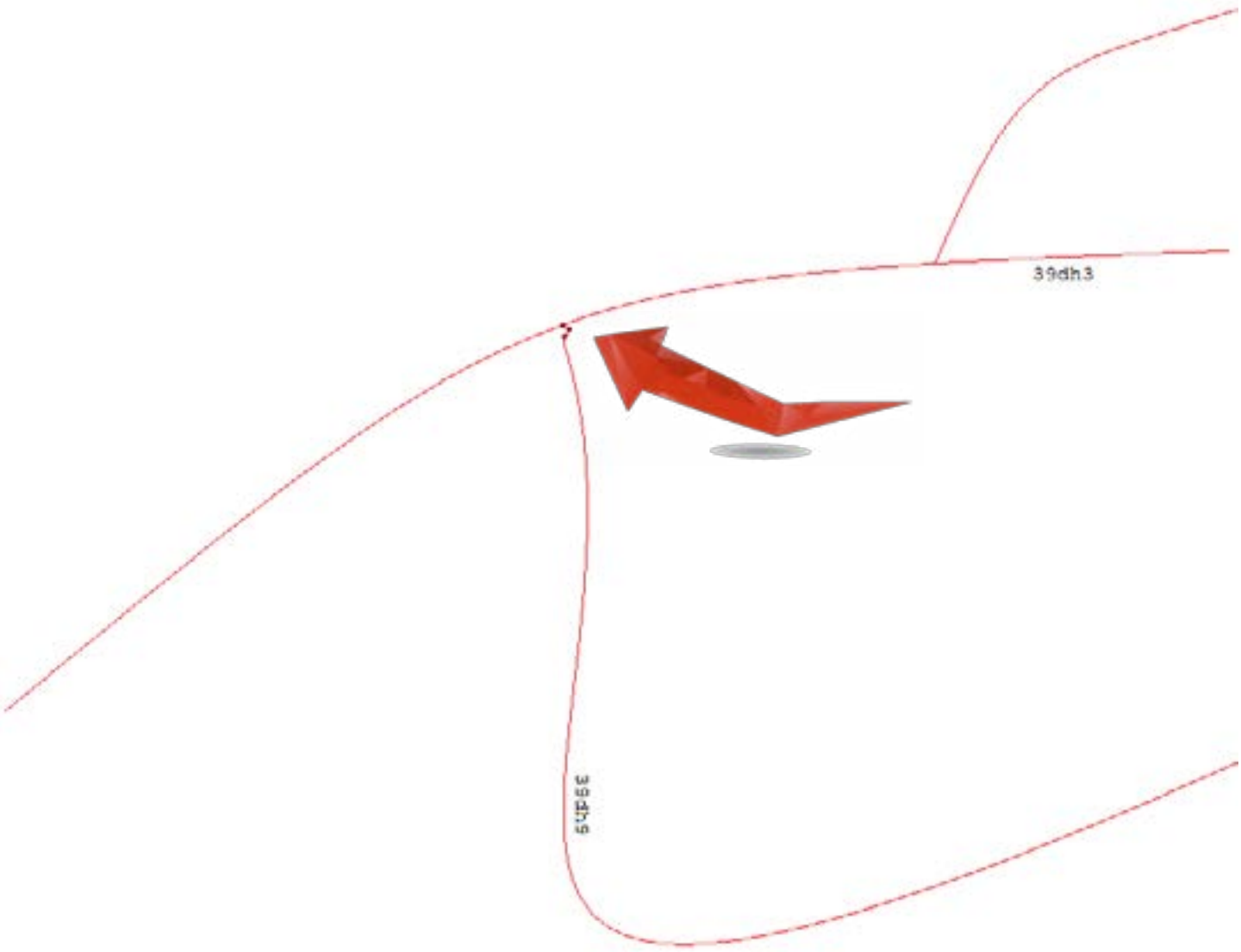


1. Duplicate points and polygons

Don't let the unique IDs fool you. These points are not different. The geometry is the same, the coordinates are the same, the attributes are the same.



Do an automated check for duplicate consecutive points with the GeometryValidator.



2. Short segments, short lines, and tiny area features

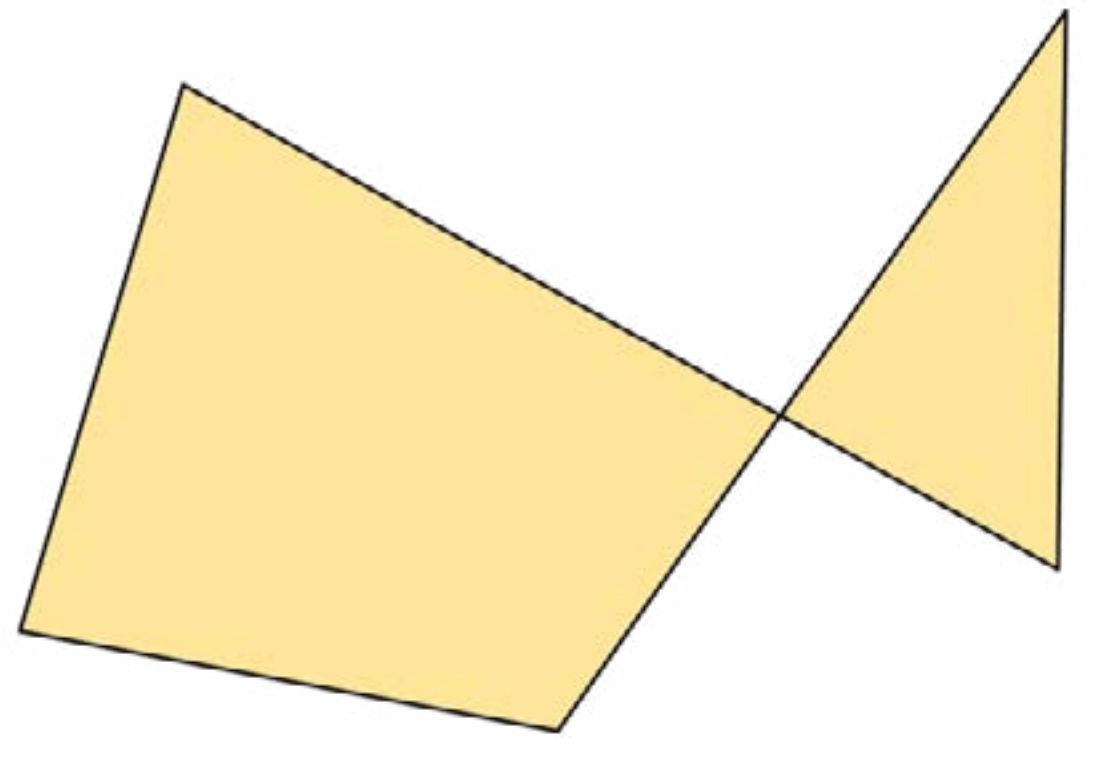
The vertices in a line or polygon are too close together, or the segments between them are too short. This is usually a red flag that something is amiss.

Automatically remove points within a given tolerance distance with the Generalizer.

3. Spikes

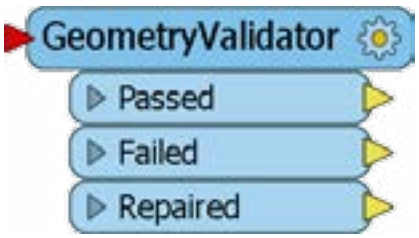
This is what it looks like when a GIS has the hiccups. An inconsistency like this is usually caused by a digitizing error.

Automatically check every pair of line segments made up of three consecutive points. If the angle between the segments is too small, the middle point is a spike and should be removed. This is a job for the SpikeRemover.

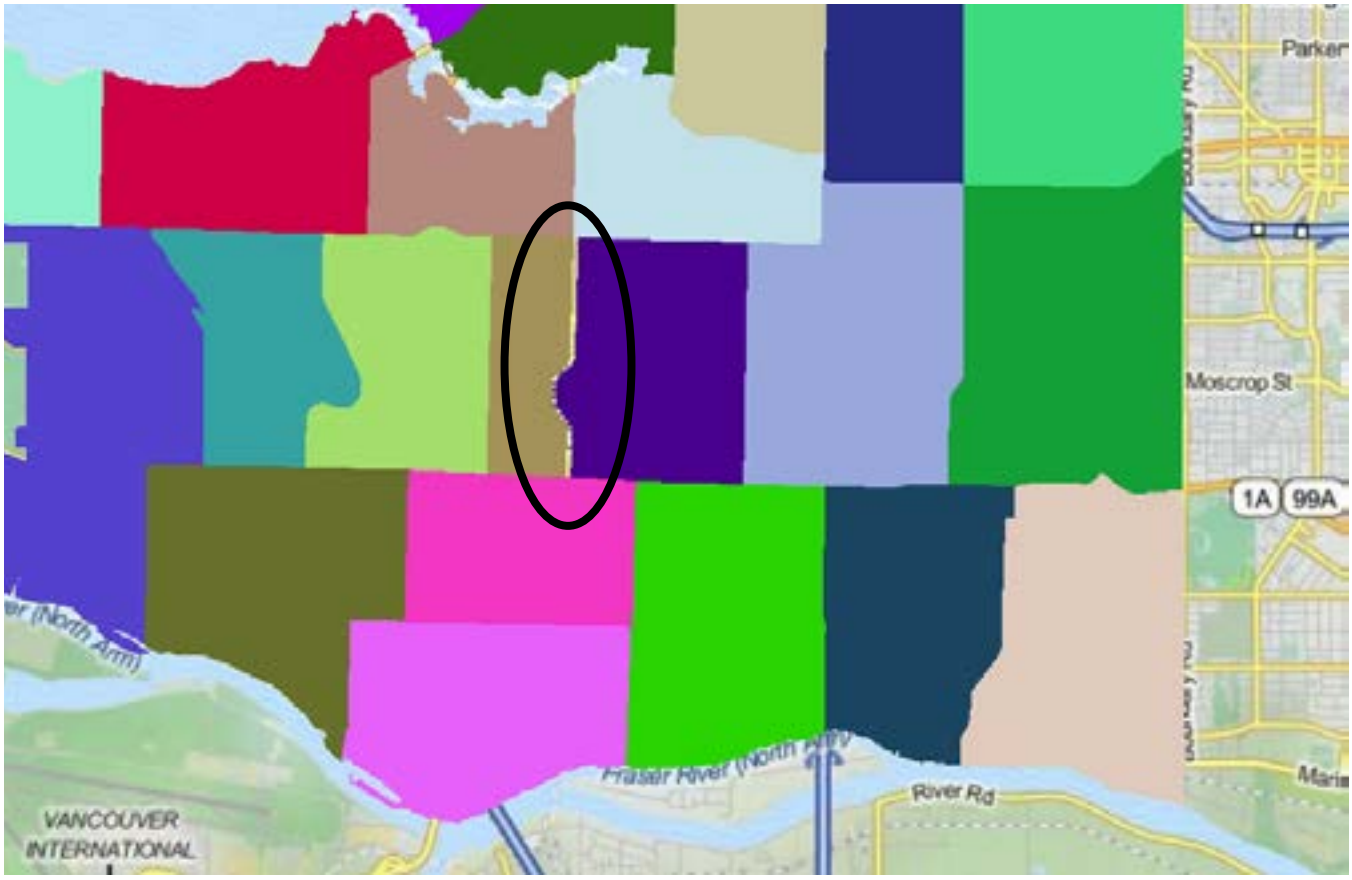


4. Self-intersections

Otherwise known as “butterfly” or “bowtie” polygons. Don’t let the word “butterfly” fool you. There is nothing pretty about this. Even after repairing, you can end up with degenerate geometries and hierarchical/nested aggregates.



Divide these into separate polygons at the intersection. For meshes, triangle strips, and triangle fans, turn them into composite surfaces. 2D self-intersections are handled by the GeometryValidator.



5. Slivers and gaps

An unintentional space between the boundaries of adjacent polygon features is called a gap. An overlap is called a sliver. Because it's tiny, barely noticeable, and feels about as good as a shard of wood jammed under your skin.

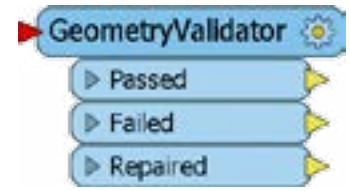
Create a triangulation of polygons and assign each overlapping region or gap to one of the adjacent areas. This is the SliverRemover.

Or, bring points together with the Snapper if they are within a certain distance of each other.

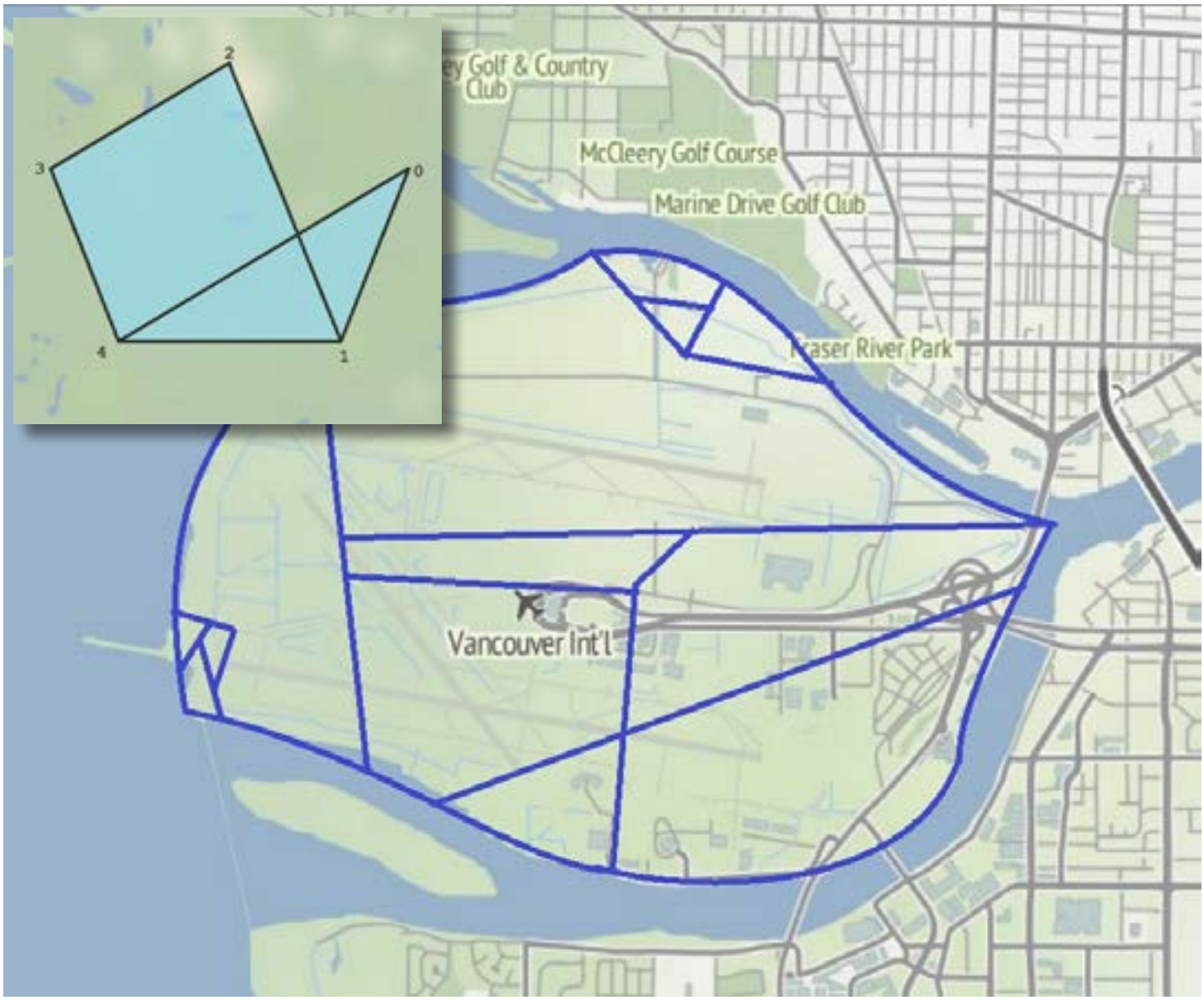
6. Unclosed polygons

Yes, because when I said 'polygon' what I really meant was 'line that looks like a polygon but doesn't actually count as one'.

An unclosed polygon, by definition, is when the start and end nodes are not equal.

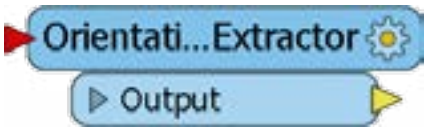


This is an example of a failure to comply to OGC standards, so the first step would be to perform an automatic check for OGC compliance using the GeometryValidator. Then, bring points together with the Snapper if they are within a certain distance of each other.



7. Order of nodes

Sometimes, polygons do not know how to count. Lines, polygons, and the relative internal/external polygons of a donut can all be affected by vertices in the wrong order or rotation direction.



Change the orientation of the offending polygon / line feature with the OrientationExtractor.

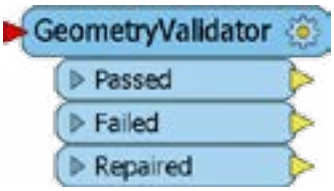
8. Overlapping polygons

This is when polygons invade each other's space and overlap with other geometry features.

Convert the intersecting parts into entirely new features with the AreaOnAreaOverlayer. Or, see #5 on dealing with Slivers and Gaps.

9. Null geometries

Null geometries contain no points. They're just empty. Nothing. Desolate.

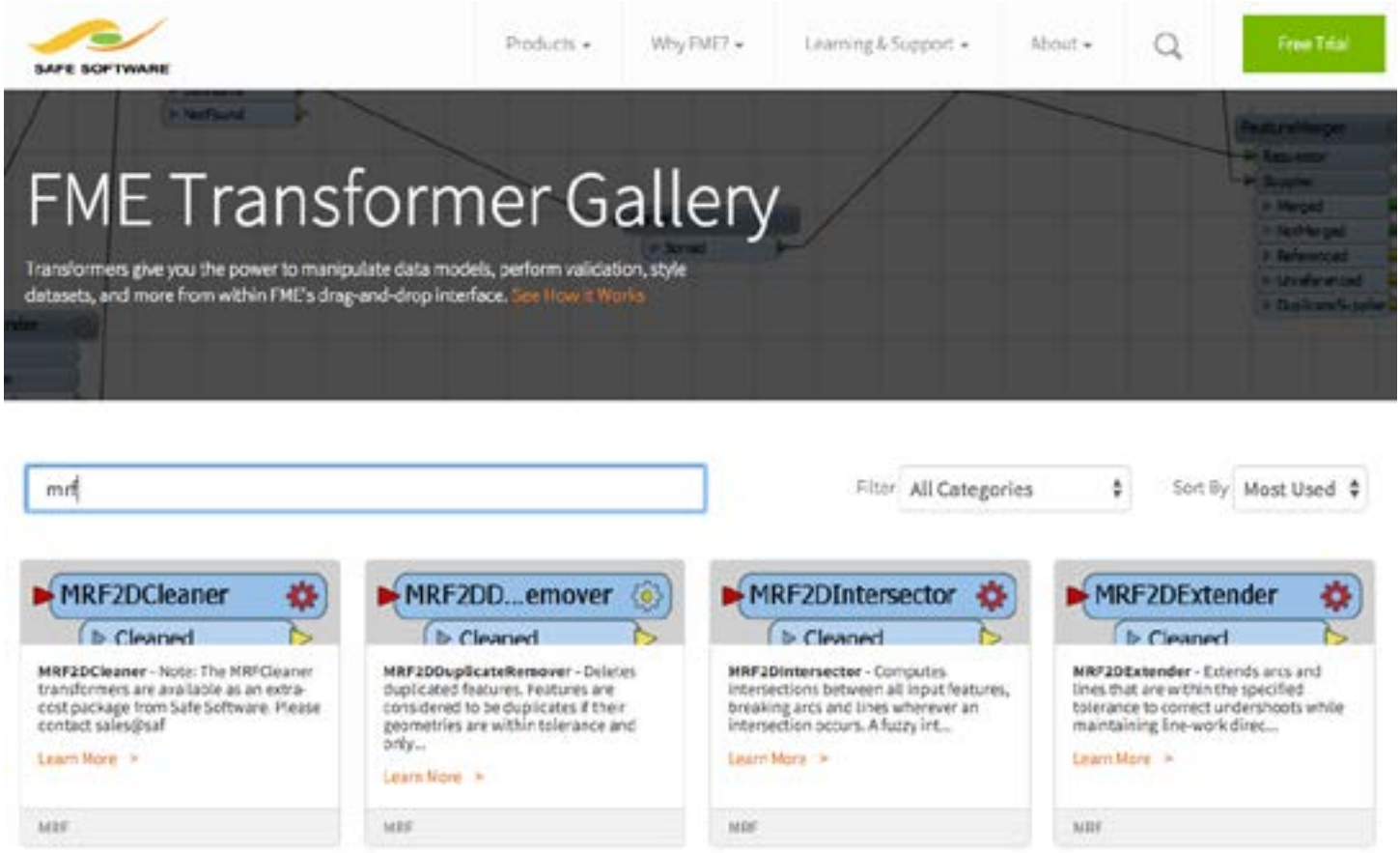


Automatically find and get rid of all null geometry parts with the GeometryValidator.

10. Undershoots and overshoots (dangles)

This is when line geometries are supposed to meet at a point, but fail to meet (undershoot) or extend beyond the meeting point (overshoot).

Remove lines that have free endpoints and have lengths smaller than a given tolerance with the MRF2DDangleRemover.



Identifying these problems is one issue, but solving them can be a whole other adventure. As you've seen in these pages, **FME** offers a lot of **transformers** to **automatically solve these problems**. Check out our Transformer Gallery for answers. Search "MRF" to see a bunch of powerful transformers that specialize in geometric operations—like those arising in CAD to GIS migrations.

What's your biggest geometry problem?

Share with us: